AD-A032 013 MARYLAND UNIV COLLEGE PARK DEPT OF COMPUTER SCIENCE F/G 12/1
AN ALGORITHM FOR COMPUTING REDUCING SUBSPACES BY BLOCK DIAGONAL--ETC(U)
OCT 76 C A BAVELY, G W STEWART N00014-76-C-0391

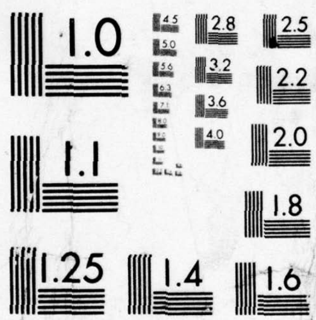
UNCLASSIFIED TR-489 NL

| OF |

AD
A032013

END

DATE
FILMED
1-77


1.0
4.5
5.0
5.6
6.3
7.1
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(2)

COMPUTER SCIENCE
TECHNICAL REPORT SERIES

UNIVERSITY OF MARYLAND
COLLEGE PARK, MARYLAND
20742

Technical Report TR-489
ONR-N00014-76-C-0391-489

October 1976

AN ALGORITHM FOR COMPUTING REDUCING SUBSPACES BY BLOCK DIAGONALIZATION

Connice A. Bavely

G. W. Stewart*

Abstract

This paper describes an algorithm for reducing a real matrix A to block diagonal form by a real similarity transformation. The columns of the transformation corresponding to a block span a reducing subspace of A, and the block is the representation of A in that subspace with respect to the basis. The algorithm attempts to control the condition of the transformation matrices, so that the reducing subspaces are well conditioned and the basis vectors are numerically independent.

*This work was supported in part by the Office of Naval Research under Contract No. N00014-76-C-0391.

AN ALGORITHM FOR COMPUTING REDUCING SUBSPACES BY BLOCK DIAGONALIZATION

Connice A. Bavely
G. W. Stewart

1. Introduction

The purpose of this report is to describe an algorithm for reducing a real matrix A of order n to a block diagonal form by a real similarity transformation. Specifically, the algorithm attempts to compute a real nonsingular matrix X such that $X^{-1}AX$ has the form

$$X^{-1}AX = B = \operatorname{diag}(B_1, B_2, \ldots, B_s) , \tag{1.1}$$

where each matrix B_i is square of order n_i.

A decomposition such as (1.1) has many applications. When the blocks B_i are small, powers of A can be economically calculated in the form

$$A^k = X \operatorname{diag}(B_1^k, B_2^k, \ldots, B_s^k) X^{-1} ,$$

and this fact can be used to simplify the computation of functions of A defined by power series (e.g. see [7]). If X is partitioned in the form

$$X = (X_1, X_2, \ldots, X_s) ,$$

where each X_i has n_i columns, then the columns of X_i form a basis for a reducing subspace of A, and B_i is the representation of A with respect to that basis. The associated spectral projector is given by $X_i X_i^{(-1)}$, where

$X_i^{(-1)}$ is formed from the corresponding rows of X^{-1} (for definitions and applications see [4]).

There are theoretical and practical limitations on how small the blocks in (1.1) can be. Theoretically, they can be no smaller than the blocks in the Jordan canonical form of A. Practically, they may have to be larger. The numerical problems associated with decompositions such as (1.1) have been examined in detail in [3]. Here we give only a brief summary.

The principal difficulty is that the Jordan form of a matrix need not be numerically well determined; very small perturbations in the matrix may cause blocks to split or coalesce. Any attempt to separate two such "nearby" blocks will result in a transformation matrix X whose columns are nearly linearly dependent, or equivalently X will be ill-conditioned in the sense that the product $\|X\|\|X^{-1}\|$ is large (here $\|\cdot\|$ denotes a suitable matrix norm). In this case, it will be impossible to form X^{-1} or solve linear systems involving X accurately [10,14]. The phenomenon is loosely associated with close eigenvalues; but there are matrices with equal eigenvalues, e.g. symmetric matrices, that can be split completely into 1×1 blocks, and there are matrices with well separated eigenvalues that cannot be split except by very ill-conditioned transformations.

Our algorithm attempts to avoid these difficulties by working only with well-conditioned transformations. If a group of eigenvalues cannot be split off into a block by a transformation whose condition observes a tolerance provided by the user, the block is enlarged until a well-conditioned reducing transformation can be found. In principle this does not insure that the final transformation will be well-conditioned, since it is formed as the product of a number of reducing transformations; however, we have found that when a matrix possesses a well-conditioned decomposition of the form (1.1), our algorithm generally finds it. And the exceptions have not so much to do with the ill-conditioning of X as with the failure of the algorithm to split the matrix completely owing to the comingling of degenerate eigenvalues with well-conditioned ones.

A good deal of work has been done on the numerically stable simplification of matrices by similarity transformations [5,6,8,12,13], most of which has been summarized in [3]. For the most part, these algorithms attempt to go farther than ours in reducing the matrix, however at considerable cost in complexity and computation. The virtues of the algorithm proposed here are its simplicity and economy. When it is required to reduce a matrix beyond what is done by our algorithm, the other techniques can be applied to the blocks produced by our algorithm. The algorithm also has the advantage that it works entirely with real matrices by the device of grouping pairs of complex conjugate eigenvalues in the same block.

In the next section of this paper the algorithm is described. In Section 3 some numerical examples are given. Programming details and a listing are given in an appendix.

2. The algorithm.

The first part of the algorithm uses orthogonal transformations to reduce the matrix A to quasi-triangular form, that is to a block upper-triangular form in which the diagonal blocks are of order at most two. The blocks of order one contain real eigenvalues of A and the blocks of order two contain complex conjugate pairs of eigenvalues. The ordering of the blocks is arbitrary, and the order can be changed by applying appropriate orthogonal transformations. Since this reduction of A can be effected by standard techniques [9,11], we may assume that A is already in quasi-triangular form.

The subsequent block diagonalization is accomplished as follows. The matrix A is partitioned in the form

$$A = \begin{pmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{pmatrix},$$

where initially A_{11} is 1×1 or 2×2 depending on the dimension of the leading diagonal block of A. An attempt is then made to find a similarity transformation X such that

$$X^{-1}AX = \begin{pmatrix} A_{11} & 0 \\ 0 & A_{22} \end{pmatrix}.$$

If such a transformation can be found and if it is not too ill-conditioned, the reduction proceeds with the submatrix A_{22}. If not, a suitable 1×1 or 2×2 diagonal block from A_{22} is located and moved by means of orthogonal transformations to the leading position of A_{22}. The block is then adjoined to A_{11} by increasing the order of A_{11} by one or two, as is appropriate, and another attempt is made to find a reducing matrix X.

The implementation of such an algorithm requires the answers to two questions.

1. How may the transformation X be computed?

2. In the event of failure, which block of A_{22} is to be incorporated into A_{11}?

We shall now answer these questions.

We seek the transformation X in the form

$$X = \begin{pmatrix} I & P \\ 0 & I \end{pmatrix}, \tag{2.1}$$

where the identity matrices are of the same orders as A_{11} and A_{22}. The inverse of X is easily seen to be

$$X^{-1} = \begin{pmatrix} I & -P \\ 0 & I \end{pmatrix}. \tag{2.2}$$

Hence

$$X^{-1}AX = \begin{pmatrix} A_{11} & A_{11}P-PA_{22}+A_{12} \\ 0 & A_{22} \end{pmatrix},$$

and the problem of determining X becomes that of solving the equation

$$A_{11}P - PA_{22} = -A_{12}. \tag{2.3}$$

Because A_{11} and A_{22} are quasi-triangular, this equation can be solved by a back-substitution algorithm of Bartels and Stewart [1], provided the eigenvalues of A_{11} and A_{22} are disjoint.

From (2.1) and (2.2) it follows that X will be ill-conditioned whenever P is large. As each element of P is generated, it is tested to see if its magnitude exceeds a bound provided by the user. If it does, the attempt to compute X *is abandoned and a new*, larger block A_{11} is formed. If no element of P exceeds the bound, the matrix X is accepted and the matrix A is deflated as described above. The transformation X is postmultiplied into a matrix that accumulates all the transformations made on the matrix A.

The process for selecting a 1×1 or 2×2 block of A_{22} to incorporate into A_{11} goes as follows. We compute the mean of those eigenvalues of A_{11} having nonnegative imaginary part. A block is chosen from A_{22} whose eigenvalue with nonnegative imaginary part is nearest this mean. This block is moved, as described above, by orthogonal transformations to the leading position in A_{22}, where it is incorporated into A_{11}. The program HQR3 [11], which can be used to obtain the initial quasi-triangular form, has a subroutine which will compute these orthogonal transformation. The transformations are of course postmultiplied into the accumulating matrix.

We summarize our algorithm in the following informal code. Further details can be found in the appendix to this paper, where a FORTRAN subroutine implementing this code is given. The code takes as input an array A of order N containing the matrix A and an array X in which the transformations are accumulated. In addition the user must provide a tolerance to bound the size of the elements of the deflating transformations. The integers L11 and L22 point to the beginnings of the current blocks A_{11} and A_{22} in the array A. The informal code should be self-explanatory. Comments are delineated by the symbol #.

```
1         : reduce A to quasitriangular form, accumulating the
              transformations in X [10];
2         : L11 = 1;
3         : loop # until the matrix is diagonalized #
3.1       :   if L11 > N then leave 3 fi;
3.2       :   L22 = L11;
3.3       :   loop # until a block has been deflated #
3.3.1     :     if L22 = L11 then # use the first 1 × 1 or 2 × 2 block #
3.3.1t.1:         M = order of the block at L11;
3.3.1t.2:         L22 = L22 + M;
3.3.1     :     else # augment A11 with a 1 × 1 or 2 × 2 block from A22 #
3.3.1e.1:         compute the mean of the eigenvalues of A11 with
                    nonnegative imaginary parts;
3.3.1e.2:         find the M × M # M = 1 or 2 # block of A22 whose eigen-
                    value with nonnegative imaginary part is nearest the
                    mean;
3.3.1e.3:         move the block to the leading position of A22 accumulating
                    the transformations in X;
3.3.1e.4:         L22 = L22 + M # which incorporates the block in A11 #;
3.3.1     :     fi;
3.3.2     :     if L22 > N then leave 3.3 fi;
3.3.3     :     attempt to split off A11 [1];
3.3.4     :     if the attempt was successful then leave 3.3 fi;
3.3.5     :     restore A12;
3.3       :   end loop;
3.4       :   if L22 ≤ N then accumulate the deflating transformation in X
                fi;
3.5       :   scale columns L11 through L22-1 of X so that they have
                2-norm unity, adjusting A accordingly;
3.6       :   L11 = L22;
3         : end loop;
```

Several comments should be made about the algorithm. First, it uses only real arithmetic, even when A has complex eigenvalues. Second, the algorithm cannot guarantee that the final transformation is well conditioned, since the bound on the elements of P restricts the condition of only the individual transformations comprising the final one. Nonetheless, we have found little tendency toward excessive growth in the condition of the transformation. Third, no attempt is made to segregate nearly equal eigenvalues initially into clusters; whether an eigenvalue splits off or not depends entirely on its determining a suitably small matrix P. This is important, since it means that the algorithm can compute well conditioned eigenvectors for multiple eigenvalues (a little help is required from rounding error; see Section 3).

The strategy for determining what eigenvalues to add to the current group has the defect that it can mix well-conditioned and ill-conditioned eigenvalues that are nearly equal, thus missing the possibility of a more complete reduction. This is not a serious problem when the blocks are small. However, if a finer resolution of the structure of the matrix is required, the techniques referenced in Section 1 may be applied to the blocks produced by our algorithm. In fact our algorithm can be regarded as a preprocessing step to reduce the problem to a size where it can be attacked by more sophisticated, but more expensive methods.

We note that the output of our algorithm may depend on the user supplied tolerance for the elements of P. In general the larger the tolerance, the smaller the blocks but the more ill conditioned the transformation. This tradeoff is an inevitable consequence of the poor determination of the structure of a matrix in the presence of errors, and we know of no algorithm that relieves the user of the necessity of making a decision of this kind.

So far as storage is concerned, the algorithm requires $2n^2$ locations to contain the matrices A and X and a number of arrays of order n. This is the same as the storages required by algorithms that compute the eigenvectors of a general matrix.

Excluding the initial reduction of A to quasitriangular form, the bulk of the computations in the algorithm occur at statement 3.3.3, where an attempt is made to solve the equation (2.3), and at statement 3.4, where the transformation is accumulated. The multiplication count for the algorithm for solving (2.3) is of order $(\ell^2 m + m^2 \ell)/2$, where ℓ is the size of A_{11} and m is the size of A_{22}. The cost of accumulating this transformation in X is of order $\ell \cdot m \cdot n$. Thus, at one extreme, if all the eigenvalues of A are real and they all can be deflated, the cost in multiplications will be of order $n^3/2$, which compares favorably with algorithms for computing the eigenvectors of A from its quasitriangular form. On the other hand, if the eigenvalues of A are real and none of them can be deflated, the algorithm will require on the order of $n^4/12$ multiplications.

Although an n^4 operation count is disturbing, there are several mitigating factors. First, we do not expect the algorithm to find many applications to matrices that cannot be substantially reduced by it, since the object of using it is to save on subsequent calculations with the matrix. Second, the count assumes that the algorithm for solving (2.3) must be executed to completion before it is found that P is unacceptably large. This is not likely; in fact because the algorithm [11] for reducing

A to quasitriangular form arranges the eigenvalues in decreasing order of magnitude, it is rather unlikely. Finally the order constant 1/12 is is modest; for n less than 60 the bound is less than $5n^3$.

3. Numerical results.

In this section we summarize the results of some numerical experiments. Two of the tests were performed with a class of test matrices generated as follows. Let J be a given matrix whose structure is known (e.g. J could be in Jordan canonical form). Let

$$H_1 = I - \frac{2}{n} ee^T ,$$

where $e = (1,1,\ldots,1)^T$, and

$$H_2 = I - \frac{2}{n} ff^T ,$$

where $f = (1,-1,\ldots,(-1)^{n-1})^T$. Note that H_1 and H_2 are symmetric and orthogonal. Let

$$S = \operatorname{diag}(1,\sigma,\sigma^2,\ldots,\sigma^{n-1})$$

where σ is a given parameter. Then we take A in the form

$$A = (H_2SH_1)J(H_2SH_1)^{-1} = (H_2SH_1)J(H_1S^{-1}H_2) . \tag{3.1}$$

The matrix A, which is cheap to compute, has the same structure as J. The transformation H_2SH_1 can be made as ill-conditioned as desired by varying σ.

In describing the results of the tests we report two numbers. The first is the maximum ρ of the scaled residuals

$$\rho_i = \frac{\|AX_i - XB_i\|_\infty}{\|A\|_\infty \|X_i\|_\infty} ,$$

where X_i is composed of the columns of X corresponding to the block B_i. Second we report $\|X^{-1}\|_\infty$. Together these numbers give an idea of how stably the reduction proceeded; for if

$$R = AX - XB$$

then, with $E = RX^{-1}$, we have that

$$(A+E)X - XB = 0 ;$$

that is B is exactly similar to $A + E$. The relative error that this perturbation represents in A is

$$\frac{\|E\|}{\|A\|} \le \frac{\|R\|\|X^{-1}\|}{\|A\|} .$$

Since $\|X\| \cong 1$, the relative error will be of the order $\rho\|X^{-1}\|$.

The first test case illustrates the ability of the algorithm to split apart nearly equal eigenvalues with independent eigenvectors. We took

$$J = \operatorname{diag}\left[1, 1-\varepsilon, 1+\varepsilon, \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}, .3, .4, .5, .6, .7\right] .$$

The algorithm was applied for various values of ε, σ, and RMAX, the bound on the size of the deflating transformations. The results are summarized in Table 1, in which the eigenvalues of A are numbered as follows:

1	2	3	4	5	6	7	8	9	10
$1+i$	$1-i$	$1+\varepsilon$	1	$1-\varepsilon$	.7	.6	.5	.4	.3

Complex eigenvalues are indicated by a circumflex.

Provided RMAX was large enough to allow a sufficiently ill conditioned transformation, all the cases were split completely. The condition of the transformation was in no case much greater than the condition of the scrambling transformation in (3.1). It is of interest to note that the algorithm was successful when $\varepsilon = 0$, that is when A has three equal eigenvalues. Mathematically, the algorithm for solving (2.3) breaks down when A_{11} and A_{22} have common eigenvalues; however, the experiments indicate that if the equal eigenvalues have independent vectors, rounding error will perturb them enough for the algorithm to work.

The second example shows the failure of the algorithm's strategy for selecting the next eigenvalue to be adjoined to A_{11}. Here J has the form

$$J = \operatorname{diag}\left[1, \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}, .3, .4, .5, .6, .7, .8\right]$$

and σ was taken to be one. Of the four eigenvalues at unity, one is perfectly conditioned, and the other three, which belong to a single Jordan block, are very ill-conditioned. To six figures the computed eigenvalues were

$$
\begin{array}{l}
1.00073 + 0.001303i \\
1.00073 - 0.001303i \\
1.00000 \\
0.99853 \\
.80000 \\
\vdots
\end{array}
$$

The pair of complex eigenvalues could not be deflated, since they were coupled to the third member of the block. But this member would not be adjoined without first adjoining the well conditioned eigenvalue at unity. Consequently, the algorithm produced a single block of order four, rather than two blocks of orders one and three. This block of order four could be reduced further by the more sophisticated techniques described in the references.

The third test case is the Frank matrix of order 12 which has appeared frequently in the literature [2,3]. The smaller eigenvalues of this matrix are extremely ill conditioned. The results of test runs on this matrix are summarized below.

RMAX	BLOCK STRUCTURE	ρ	$\|X^{-1}\|_\infty$
10	1,2,3,4,5,(6,7,8,9,10,11,12)	2.9×10^{-7}	93.6
50	1,2,3,4,5,6,(7,8,9,10,11,12)	2.9×10^{-7}	2920.2
100	1,2,3,4,5,6,(7,8,9,10,11,12)	2.9×10^{-7}	2920.2
1000	1,2,3,4,5,6,(7,8,9,10,11,12)	2.9×10^{-7}	2920.2

The algorithm performed much as expected, separating the larger eigenvalues and grouping the smaller eigenvalues together. This grouping is consistent with the precision of the computation.

The final test case is included because the failure of our algorithm to decompose it reveals the shaky foundations of a fairly common numerical practice. Specifically we generated the companion matrix of the polynomial given in [14, p. 74]. Since the zeros of this polynomial are not very ill conditioned, we were surprised when the algorithm failed to split off so much as one. Some further computations revealed that the matrix of left eigenvectors (the inverse Vandermode of the zeros) had rows of order 10^6 - 10^8. This explains the failure to reduce the matrix. More important, though, it shows that the eigenvalues of the companion matrix are much more ill conditioned than the zeros of the polynomial and suggests that the practice of using eigenvalue routines to find zeros of polynomials can result in an unnecessary loss in accuracy.

σ	ε	RMAX	BLOCK STRUCTURE	ρ	$\|X^{-1}\|_\infty$
1.0	10^{-1}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	2.1×10^{-7}	7.7
	10^{-5}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.4×10^{-7}	6.7
	0.0	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.1×10^{-7}	8.0
10.0	10^{-1}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.5×10^{-7}	25.6
	10^{-5}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.8×10^{-7}	25.4
	0.0	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.6×10^{-7}	24.2
100.0	10^{-1}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.1×10^{-7}	187.9
	10^{-5}	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.3×10^{-7}	196.6
	0.0	10.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.2×10^{-7}	157.9
1000.0	10^{-1}	10.	$(\hat{1},\hat{2},4,3,5),(6,7,8,9),10$	1.1×10^{-7}	326.6
		100.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	1.0×10^{-8}	1537.9
	10^{-5}	10.	$(\hat{1},\hat{2},4,3,5),(6,7,8,9),10$	7.8×10^{-8}	333.8
		100.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	7.8×10^{-8}	1559.1
	0.0	10.	$(\hat{1},\hat{2},3),4,5,(6,7,8,9),10$	8.2×10^{-8}	394.1
		100.	$(\hat{1},\hat{2}),3,4,5,6,7,8,9,10$	8.2×10^{-8}	1355.7

Table 4.1

References

1. R. H. Bartels and G. W. Stewart, Algorithm 432, The solution of the matrix equation AX - XB = C, Comm. Assoc. Comput. Mach. (1972) 820-826.

2. R. T. Gregory and D. L. Karney, A Collection of Matrices for Testing Computational Algorithms, Wiley-Interscience, New York (1969).

3. G. H. Golub and J. H. Wilkinson, Ill-conditioned eigensystems and the computation of the Jordan canonical form, Stanford University Report STAN-CS-75-478 (1975).

4. T. Kato, Perturbation Theory for Linear Operators, Springer Verlag, New York (1966).

5. V. N. Kublanovskaya, On a method of solving the complete eigenvalue problem for a degenerate matrix, Z. Vycisl. Mat. Mat. Fiz. 6 (1966) 611-620; translation in USSR Comp. Math. Math. Phys. 6 (1968) 1-14.

6. B. Kågström and Axel Ruhe, An algorithm for numerical computation of the Jordan normal form of a complex matrix, University of Umeå, UMINF-51.74, Sweden (1977).

7. C. B. Moler and C. F. Van Loan, Nineteen ways to compute the exponential of a matrix, Cornell University Computer Science Technical Report TR 76-283 (1976).

8. A. Ruhe, An algorithm for numerical determination of the structure of a general matrix, BIT 10 (1970), 196-216.

9. B. T. Smith, J. M. Boyle, J. J. Dongarra, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, Matrix Eigensystem Routines, EISPACK Guide (Second Edition), Springer Verlag, New York (1976).

10. G. W. Stewart, Introduction to Matrix Computations, Academic Press, New York (1973).

11. ____________, Algorithm 506: HQR3 and EXCHNG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix, ACM Trans. Math. Software 3 (1976) 275-280.

12. J. M. Varah, Rigorous machine bounds for the eigensystem of a general complex matrix. Math. Comp. 22 (1968) 293-801.

13. ______________, Computing invariant subspaces of a general matrix when the eigensystem is poorly conditioned, Math. Comp. 24 (1970) 137-149.

14. J. H. Wilkinson, Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, New Jersey (1963).

Appendix

PROGRAMMING DETAILS AND PROGRAM LISTING

A1. Usage.

The calling sequence for BDIAG is

CALL BDIAG(A,LDA,N,EPSHQR,RMAX,ER,EI,TYPE,BS,X,LDX,FAIL).

The parameters in the calling sequence are (starred parameters are altered by the subroutine)

*A(LDA,N)	an array that initially contains the $N \times N$ matrix to be reduced. On return A contains the reduced block diagonal matrix.
LDA	the leading dimension of A.
N	the order of the matrices A and X.
EPSHQR	a real variable containing a convergence criterion for the subroutines HQR3 and EXCHNG [11].
RMAX	a real variable containing a bound on the absolute values of the elements in the reducing matrices.
*ER(N)	a real array containing the real parts of the eigenvalues of A.
*EI(N)	a real array containing the imaginary parts of the eigenvalues.
*TYPE(N)	an integer array whose i-th entry is 0 if the i-th eigenvalue is real 1 if the i-th eigenvalue is complex with positive imaginary part 2 if the i-th eigenvalue is complex with negative imaginary part -1 if the i-th eigenvalue could not be computed
*BS(N)	a singly subscripted array that contains information on the block structure of the matrix returned by the program. If there is a block of order K at A(L,L), then BS(L),BS(L+1),...BS(L+K-1) contain the integers K,-(K-1),...,-1. Thus a positive entry of K indicates the start of a block of order K.

*X(LDX,N)	an array into which the reducing transformations are accumulated.
LDX	the leading dimension of X.
FAIL	a logical variable which is false on a normal return and is true on return if an error has occurred.

The program requires the programs ORTHES [9], ORTRAN [9], HQR3 [11], EXCHNG [11], SHRSLV [1], DAD and SPLIT [11].

A suitable choice for the parameter EPSHQR is the rounding unit of the computer on which the program is being run; i.e. if one is working in a precision of about t decimal figures EPSHQR should be of order 10^{-t}.

A2. Programming details.

In this section we shall describe some of the details of the implementation of the algorithm. Throughout this section we refer to the outline of the algorithm in Section 2.

Statement 1. This is accomplished by the EISPACK routines ORTHES and ORTRAN [9] and the QR routine HQR3 [11].

Statement 2. L11 points to the leading position of the current block A11, which is of order DA11.

Statement 3. This is the main loop of the program. It ends when L11 > N, indicating that there are no further blocks to deflate.

Statement 3.2. L22 points to the leading position of the current block A22, which is of order DA22.

Statement 3.3. In this loop A_{11} is repeatedly augmented until it can be deflated or until A_{22} is void.

Statements 3.3.1t. A_{11} is initially void. Here it is taken to be the 1×1 or 2×2 block starting at L11.

Statements 3.3.1e. This is the search for a 1×1 or 2×2 block described in Section 2.

Statement 3.3.1e.3. The subroutine EXCHNG [11] is used repeatedly to move the block just located to the beginning of A_{22}. After each exchange of a complex block SPLIT [11] is called to recompute its eigenvalues and to see if, owing to rounding error, it can be split into a pair of real eigenvalues.

Statement 3.3.2. Because A_{22} is void, A_{11} is effectively deflated.

Statement 3.3.3. The matrix A_{12} is saved below the lower subdiagonal of A, in case the attempt to deflate A_{11} is unsuccessful. Since the routine SHRSLV [1], which computes the deflating transformation, requires A_{11} to be lower Hessenberg, the subroutine DAD is called to transform A_{11} and A_{12}. SHRSLV has been modified to return with a signal if some element of the deflating transformation exceeds the bound RMAX. Otherwise the matrix P that determines the transformation overwrites A_{12}. DAD is once again called to restore A_{11} to its original form.

Statement 3.3.5. The submatrix A_{22}, which was overwritten by SHRSLV, must be restored before another attempt to deflate is made.

Statement 3.4. Only if A_{22} is not void was a deflating transformation generated.

Statement 3.6. Set L11 to point to the next submatrix before continuing.

A3. Program listing.

```
      SUBROUTINE BDIAG (A,LDA,N,EPSHQR,RMAX,ER,EI,TYPE,BS,X,LDX,FAIL)
C
C
      INTEGER LDA, LDX, N, TYPE(N), BS(N)
      REAL A(LDA,N), ER(N), EI(N), X(LDX,N), EPSHQR, RMAX
      LOGICAL FAIL
C
C
C     BDIAG REDUCES A MATRIX A TO BLOCK DIAGONAL FORM BY FIRST
C     REDUCING IT TO QUASI-TRIANGULAR FORM BY HQR3 AND THEN BY
C     SOLVING THE MATRIX EQUATION  -A11*P+P*A22=A12 TO INTRODUCE ZEROS
C     ABOVE THE DIAGONAL.  THE PARAMETERS IN THE CALLING SEQUENCE
C     ARE (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE):
C
C
C     *A        AN ARRAY THAT INITIALLY CONTAINS THE N X N MATRIX
C               TO BE REDUCED. ON RETURN, A CONTAINS THE REDUCED
C               BLOCK DIAGONAL MATRIX.
C
C      LDA      THE LEADING DIMENSION OF ARRAY A.
C
C      N        THE ORDER OF THE MATRICES A AND X.
C
C      EPSHQR   THE CONVERGENCE CRITERION FOR SUBROUTINE HQR3.
C
C      RMAX     THE MAXIMUM SIZE ALLOWED FOR ANY ELEMENT OF THE
C               TRANSFORMATIONS.
C
C     *ER       A SINGLY SUBSCRIPTED REAL ARRAY CONTAINING THE REAL
C               PARTS OF THE EIGENVALUES.
C
C     *EI       A SINGLY SUBSCRIPTED REAL ARRAY CONTAING THE IMAGINARY
C               PARTS OF THE EIGENVALUES.
C
C     *TYPE     A SINGLY SUBSCRIPTED INTEGER ARRAY WHOSE I-TH ENTRY IS
C               0  IF THE I-TH EIGENVALUE IS REAL
C               1  IF THE I-TH EIGENVALUE IS COMPLEX WITH POSITIVE
C                  IMAGINARY PART
C               2  IF THE I-TH EIGENVALUE IS COMPLEX WITH NEGATIVE
C                  IMAGINARY PART
C              -1 IF THE I-TH EIGENVALUE COULD NOT BE COMPUTED
C
C     *BS       A SINGLY SUBSCRIPTED INTEGER ARRAY THAT CONTAINS BLOCK
C               STRUCTURE INFORMATION.  IF THERE IS A BLOCK OF ORDER
C               K STARTING AT A(L,L) IN THE OUTPUT MATRIX A, THEN
C               BS(L) CONTAINS THE POSITIVE INTEGER K, BS(L+1) CONTAINS
C               -(K-1), BS(LA+2) = -(K-2), ..., BS(L+K-1) = -1.
C               THUS A POSITIVE INTEGER IN THE L-TH ENTRY OF BS
C               INDICATES A NEW BLOCK OF ORDER BS(L) STARTING AT A(L,L).
C
C     *X        AN ARRAY INTO WHICH THE REDUCING TRANSFORMATIONS
C               ARE TO BE MULTIPLIED.
C
C      LDX      THE LEADING DIMENSION OF ARRAY X.
C
C     *FAIL     A LOGICAL VARIABLE WHICH IS FALSE ON NORMAL RETURN AND
C               TRUE IF THERE IS ANY ERROR IN BDIAG.
C
C
C
C
C     BDIAG USES SUBROUTINES ORTHES, ORTRAN, HQR3, EXCHNG, SHRSLV, DAD, AND SPLIT.
C
C
      INTEGER DA11, DA22, I, J, K, KM1, KM2, L, LEAVE, LOOP,
     1 L11, L22, L22M1, NK
      REAL C, CAV, D, E1, E2, RAV,  SC, TEMP
C
C
      FAIL = .TRUE.
C
C     CONVERT A TO UPPER HESSENBERG FORM.
C
      CALL ORTHES (LDA,N,1,N,A,ER)
```

```
      CALL ORTRAN (LDA,N,1,N,A,ER,X)
C
C     CONVERT A TO QUASI-UPPER TRIANGULAR FORM BY QR METHOD.
C
      CALL HQR3 (A,X,N,1,N,EPSHQR,ER,EI,TYPE,LDA,LDX)
C
C     CHECK TO SEE IF HQR3 FAILED IN COMPUTING ANY EIGENVALUE
C
C
      DO 5 I = 1,N
        IF (TYPE(I).EQ.-1) GO TO 900
    5 CONTINUE
C
C     REDUCE A TO BLOCK DIAGONAL FORM
C
C     SEGMENT A INTO 4 MATRICES: A11, A DA11 X DA11 BLOCK
C     WHOSE (1,1)-ELEMENT IS AT A(L11,L11); A22, A DA22 X DA22
C     BLOCK WHOSE (1,1)-ELEMENT IS AT A(L22,L22);  A12,
C     A DA11 X DA22 BLOCK WHOSE (1,1)-ELEMENT IS AT A(L11,L22);
C     AND A21, A DA22 X DA11 BLOCK = 0 WHOSE (1,1)-
C     ELEMENT IS AT A(L22,L11).
C
C
C
C     THIS LOOP USES L11 AS LOOP INDEX AND SPLITS OFF A BLOCK
C     STARTING AT A(L11,L11).
C
C
      L11 = 1
   10 ASSIGN 550 TO LOOP
        ASSIGN 600 TO LEAVE
        IF (L11 .GT. N) GO TO LEAVE
        L22 = L11
C
C       THIS LOOP USES DA11 AS LOOP VARIABLE AND ATTEMPTS TO SPLIT
C       OFF A BLOCK OF SIZE DA11 STARTING AT A(L11,L11)
C
  100   ASSIGN 350 TO LOOP
          IF (L22 .NE. L11) GO TO 110
            DA11 = TYPE(L11) + 1
            L22 = L11 + DA11
            L22M1 = L22 - 1
          GO TO 290
  110     CONTINUE
C
C
C           COMPUTE THE AVERAGE OF THE EIGENVALUES IN A11
C
            RAV = 0.
            CAV = 0.
            DO 120 I = L11,L22M1
              RAV = RAV + ER(I)
              CAV = CAV + ABS(EI(I))
  120       CONTINUE
            RAV = RAV / FLOAT(DA11)
            CAV = CAV / FLOAT(DA11)
C
C           LOOP ON EIGENVALUES OF A22 TO FIND THE ONE CLOSEST TO THE AVERAGE.
C
            D = (RAV-ER(L22))**2 + (CAV-EI(L22))**2
            K = L22
            L = L22 + TYPE(L22) + 1
  130       ASSIGN 145 TO LOOP
              IF (L.GT.N) GO TO 150
              C = (RAV-ER(L))**2 + (CAV-EI(L))**2
              IF (C.GE.D) GO TO 140
                K = L
                D = C
  140         CONTINUE
              L = L + TYPE(L) + 1
  145       GO TO 130
  150       CONTINUE
C
C
C           LOOP TO MOVE THE EIGENVALUE JUST LOCATED
C           INTO FIRST POSITION OF BLOCK A22.
```

```
C
      ASSIGN 280 TO LOOP
         IF (TYPE(K).NE.0) GO TO 200
C
C        THE BLOCK WE'RE MOVING TO ADD TO A11 IS A 1 X 1
C
         NK = 1
  160    CONTINUE
         IF (K .EQ. L22) GO TO 280
         KM1 = K - 1
         IF (TYPE(KM1).EQ.0) GO TO 190
C
C        WE'RE SWAPPING THE CLOSEST BLOCK WITH A 2 X 2
C
         KM2 = K - 2
         CALL EXCHNG (A,X,N,KM2,2,1,EPSHQR,FAIL,LDA,LDX)
         IF (FAIL) GO TO 900
C
C        TRY TO SPLIT THIS BLOCK INTO 2 REAL EIGENVALUES
C
         CALL SPLIT (A,X,N,KM1,E1,E2,LDA,LDX)
         IF (A(K,KM1).EQ.0.) GO TO 170
C
C        BLOCK IS STILL COMPLEX.
C
         TYPE(KM2) = 0
         TYPE(KM1) = 1
         TYPE(K) = 2
         ER(KM2) = ER(K)
         EI(KM2) = 0.
         ER(K) = E1
         ER(KM1) = E1
         EI(KM1) = E2
         EI(K) = -E2
         GO TO 180
C
C        COMPLEX BLOCK SPLIT INTO TWO REAL EIGENVALUES.
C
  170    CONTINUE
         TYPE(KM1) = 0
         TYPE(KM2) = 0
         ER(KM2) = ER(K)
         ER(KM1) = E1
         ER(K) = E2
         EI(KM2) = 0.
         EI(KM1) = 0.
  180    K = KM2
         IF (K .LE. L22) GO TO 280
         GO TO 160
C
C
C        WE'RE SWAPPING THE CLOSEST BLOCK WITH A 1 X 1.
C
  190    CONTINUE
         CALL EXCHNG (A,X,N,KM1,1,1,EPSHQR,FAIL,LDA,LDX)
         IF (FAIL) GO TO 900
         TEMP = ER(K)
         ER(K ) = ER(KM1)
         ER(KM1) = TEMP
         K = KM1
         IF (K .LE. L22) GO TO 280
         GO TO 160
C
C        THE BLOCK WE'RE MOVING TO ADD TO A11 IS A 2 X 2.
C
  200    CONTINUE
         NK = 2
  210    CONTINUE
         IF (K .EQ. L22) GO TO 280
         KM1 = K - 1
         IF (TYPE(KM1) .EQ. 0) GO TO 240
C
C        WE'RE SWAPPING THE CLOSEST BLOCK WITH A 2 X 2 BLOCK.
C
         KM2 = K - 2
         CALL EXCHNG (A,X,N,KM2,2,2,EPSHQR,FAIL,LDA,LDX)
```

```
      IF (FAIL) GO TO 900
C
C     TRY TO SPLIT SWAPPED BLOCK INTO TWO REALS.
C
      CALL SPLIT (A,X,N,K,E1,E2,LDA,LDX)
      ER(KM2) = ER(K)
      ER(KM1) = ER(K+1)
      EI(KM2) = EI(K)
      EI(KM1) = EI(K+1)
      IF (A(K+1,K) .EQ. 0.) GO TO 220
C
C     STILL COMPLEX BLOCK.
C
      ER(K) = E1
      ER(K+1) = E1
      EI(K) = E2
      EI(K+1) = -E2
      GO TO 230
C
C     TWO REAL ROOTS.
C
  220 CONTINUE
      TYPE(K) = 0
      TYPE(K+1) = 0
      ER(K) = E1
      ER(K+1) = E2
      EI(K) = 0.
      EI(K+1) = 0.
  230 CONTINUE
      K = KM2
      IF (K.EQ.L22) GO TO 260
      GO TO 210
C
C     WE'RE SWAPPING THE CLOSEST BLOCK WITH A 1 X 1.
C
  240 CONTINUE
      CALL EXCHNG (A,X,N,KM1,1,2,EPSHQR,FAIL,LDA,LDX)
      IF (FAIL) GO TO 900
      TYPE (KM1) = 1
      TYPE(K) = 2
      TYPE(K+1) = 0
      ER(K+1) = ER(KM1)
      ER(KM1) = ER(K)
      EI(KM1) = EI(K)
      EI(K) = EI(K+1)
      EI(K+1) = 0.
      GO TO 250
C
  250 CONTINUE
      K = KM1
      IF (K.EQ.L22) GO TO 260
      GO TO 210
C
C     TRY TO SPLIT RELOCATED COMPLEX BLOCK.
C
  260 CONTINUE
      CALL SPLIT (A,X,N,K,E1,E2,LDA,LDX)
      IF (A(K+1,K).EQ.0.) GO TO 270
C
C     STILL COMPLEX.
C
      ER(K) = E1
      ER(K+1) = E1
      EI(K) = E2
      EI(K+1) = -E2
      GO TO 280
C
C     SPLIT INTO TWO REAL EIGENVALUES.
C
  270 CONTINUE
      TYPE(K) = 0
      TYPE(K+1) = 0
      ER(K) = E1
      ER(K+1) = E2
      EI(K) = 0.
      EI(K+1) = 0.
```

```
C
  280          CONTINUE
               DA11 = DA11 + NK
               L22 = L11 + DA11
               L22M1 = L22 - 1
  290       CONTINUE
            ASSIGN 400 TO LEAVE
            IF (L22 .GT. N) GO TO LEAVE
C
C           ATTEMPT TO SPLIT OFF A BLOCK OF SIZE DA11.
C
            DA22 = N - L22 + 1
C
C           SAVE A12 IN ITS TRANSPOSE FORM IN BLOCK A21.
C
            DO 300 J = L11,L22M1
               DO 300 I = L22,N
                  A(I,J) = A(J,I)
  300       CONTINUE
C
C
C           CONVERT A11 TO LOWER QUASI-TRIANGULAR AND MULTIPLY IT BY -1 AND CHANGE
C           A12 APPROPRIATELY (FOR SOLVING -A11*P+P*A22=A12).
C
            CALL DAD (A,LDA,L11,L22M1,L11,N,1.,0)
            CALL DAD (A,LDA,L11,L22M1,L11,L22M1,-1.,1)
C
C           SOLVE -A11*P + P*A22 = A12.
C
            CALL SHRSLV (A(L11,L11),A(L22,L22),A(L11,L22),
     1         DA11,DA22,LDA,LDA,LDA,RMAX,FAIL)
            ASSIGN 400 TO LEAVE
            IF (.NOT. FAIL) GO TO LEAVE
C
C           CHANGE A11 BACK TO UPPER QUASI-TRIANGULAR.
C
            CALL DAD (A,LDA,L11,L22M1,L11,L22M1,1.,1)
            CALL DAD (A,LDA,L11,L22M1,L11,L22M1,-1.,0)
C
C           WAS UNABLE TO SOLVE FOR P - TRY AGAIN
C
C
C
C           MOVE SAVED A12 BACK INTO ITS CORRECT POSITION.
C
            DO 310 J=L11,L22M1
               DO 310 I = L22,N
                  A(J,I) = A(I,J)
                  A(I,J) = 0.
  310       CONTINUE
C
C
  350    GO TO 100
  400    CONTINUE
C
C        CHANGE SOLUTION TO P TO PROPER FORM.
C
         IF (L22 .GT. N) GO TO 440
            CALL DAD (A,LDA,L11,L22M1,L11,N,1.,0)
            CALL DAD (A,LDA,L11,L22M1,L11,L22M1,-1.,1)
C
C           MULTIPLY TRANSFORMATION INTO X.
C
C           ONLY COLUMNS L22 THRU N ARE AFFECTED.
C
            DO 410 J = L22,N
               DO 410 I = 1,N
                  DO 410 K = L11,L22M1
                     X(I,J) = X(I,J) + X(I,K) * A(K,J)
  410       CONTINUE
C
C           ZERO OUT A12 FOR EASE IN HANDLING.
C
            DO 420 J = L22,N
               DO 420 I = L11,L22M1
                  A(I,J) = 0.
  420       CONTINUE
```

```
C
C     ZERO OUT TRIANGULAR BLOCK BELOW DIAGONAL.
C
           DO 430 J = L11,L22M1
              DO 430 I = L22,N
                 A(I,J) = 0.
  430      CONTINUE
C
C     SCALE THOSE COLUMNS OF X THAT WON'T BE ALTERED AGAIN TO UNITY.
C     CHANGE A11 APPROPRIATELY.
C
  440   CONTINUE
        DO 500 J = L11,L22M1
           SC = 0.
           DO 450 I = 1,N
              SC = SC + (X(I,J))**2
  450      CONTINUE
           SC = SQRT(SC)
           DO 460 I = 1,N
              X(I,J) = X(I,J) / SC
  460      CONTINUE
           DO 470 I = L11,L22M1
              A(I,J) = A(I,J) / SC
  470      CONTINUE
           DO 480 I = L11,L22M1
              A(J,I) = A(J,I) * SC
  480      CONTINUE
  500   CONTINUE
C
C     STORE BLOCK SIZE IN ARRAY BS.
C
        BS(L11) = DA11
        J = DA11 - 1
        IF (J .EQ. 0) GO TO 520
           DO 510 I = 1,J
              BS(L11+I) = -(DA11-I)
  510      CONTINUE
  520   CONTINUE
        L11 = L22
  550 GO TO 10
  600 CONTINUE
      FAIL = .FALSE.
      RETURN
C
C     ERROR RETURN.
C
  900 CONTINUE
      FAIL = .TRUE.
      RETURN
      END
```

```
      SUBROUTINE DAD (A, NA, I1, I2, J1, J2, R,ISW)
C
C
C           IF ISW = 0, SUBROUTINE DAD
C           COMPUTES D*A WHERE D IS THE MATRIX WITH ONES DOWN
C           THE MINOR DIAGONAL AND A IS THE INPUT MATRIX.
C           IF ISW = 1, IT COMPUTES THE PRODUCT A*D.
C           IT COMPUTES THIS PRODUCT FOR ROWS I1 THRU I2 AND COL-
C           UMNS J1 THRU J2. IT ALSO MULTIPLIES EACH ELEMENT OF
C           THE PRODUCT WITH THE CONSTANT R. NA IS THE FIRST
C           DIMENSION OF THE MATRIX A.  THE PRODUCT
C           OVERWRITES THE SPECIFIED LOCATIONS OF MATRIX A.
C
      REAL A(NA,1), R
      IF (ISW .EQ. 1) GO TO 200
      IF (I1 .EQ. I2) GO TO 150
C
      NRD2 = IFIX((I2 - I1 + 1) / 2    )
      DO 100 J = J1,J2
         DO 50 IP1 = 1,NRD2
            I = IP1 - 1
            TEMP = A(I1+I,J)
            A(I1+I,J) = A(I2-I,J) * R
            A(I2-I,J) = TEMP * R
50       CONTINUE
100   CONTINUE
      IF (MOD(I2-I1,2) .EQ. 1) RETURN
      I = I1 + NRD2
      DO 110 J=J1,J2
         A(I,J) = A(I,J) * R
110   CONTINUE
      RETURN
150   CONTINUE
      DO 160 J = J1,J2
         A(I1,J) = A(I1,J) * R
160   CONTINUE
      RETURN
C
C
C           COMPUTES THE PRODUCT AD WHERE D IS AS ABOVE.
C
C
200   CONTINUE
      IF (J1 .EQ. J2) GO TO 350
      NCD2 = IFIX((J2 - J1 + 1) / 2    )
      DO 300 JP1 = 1,NCD2
         J = JP1 - 1
         DO 250 I = I1,I2
            TEMP = A(I,J1+J)
            A(I,J1+J) = A(I,J2-J) * R
            A(I,J2-J) = TEMP * R
250      CONTINUE
300   CONTINUE
      IF ( MOD(J2-J1,2) .EQ. 1) RETURN
      J = J1 + NCD2
      DO 310 I=I1,I2
         A(I,J) = A(I,J) * R
310   CONTINUE
      RETURN
350   CONTINUE
      DO 360 I = I1,I2
         A(I,J1) = A(I,J1) * R
360   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SHRSLV (A,B,C,M,N,NA,NB,NC,RMAX,FAIL)
C
C     SHRSLV IS A FORTRAN IV SUBROUTINF TO SOLVE THF REAL MATRIX
C     EQUATION AX + XB = C, WH RE A IS IN LOWER REAL SCHUR FORM
C     AND B IS IN UPPER REAL SCHUR FORM.  SHRSLV USES THE AUX-
C     ILLIARY SUBROUTINE SYSSLV, WHICH IT COMMUNICATES WITH THROUGH
C     THE COMMON BLOCK SLVBLK.  THE PARAMETERS IN THE CALLING
C     SEQUENCE ARE
C
C     A       A DOUBLY SUBSCRIPTED ARRAY CONTAING THE MATRIX A IN
C             LOWER SCHUR FORM
C
C     B       A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE MATRIX B
C             IN UPPER REAL SCHUR FORM
C
C     C       A DOUBLY SUBSCRIPTED ARRAY CONTAINING THE MATRIX C.
C
C     M       THE ORDER OF THE MATRIX A
C
C     N       THE ORDER OF THE MATRIX B
C
C     NA      THE FIRST DIMENSION OF THF ARRAY A
C
C     NB      THE FIRST DIMENSION OF THF ARRAY B
C
C     NC      THE FIRST DIMENSION OF THF ARRAY C
C
C     RMAX    MAXIMUM ALLOWED SIZE OF ANY ELEMENT OF THF TRANSFORMATION
C
C     FAIL  INDICATES IF SHRSLV FAILED
C
C
C
      INTEGER M,N,NA,NB,NC,K,KM1,DK,KK,L,LM1,DL,LL,I,IB,J,JA,NSYS
      REAL A(NA,1), B(NB,1), C(NC,1), T, P
      LOGICAL FAIL, SING
      COMMON/SLVBLK/ T(5,5),P(5),NSYS,SING
      FAIL = .TRUE.
      L = 1
   10 LM1 = L - 1
      DL = 1
      IF (L .EQ. N) GO TO 15
      IF (B(L+1,L) .NE. 0.) DL = 2
   15 LL = L+DL-1
      IF (L .EQ. 1) GO TO 30
      DO 20 J = L,LL
         DO 20 I = 1,M
            DO 20 IB = 1,LM1
               C(I,J) = C(I,J) - C(I,IB)*B(IB,J)
   20 CONTINUE
   30 K = 1
   40 KM1 = K - 1
      DK = 1
      IF (K .EQ. M) GO TO 45
      IF (A(K,K+1) .NE. 0.) DK = 2
   45 KK = K+DK-1
      IF (K .EQ. 1) GO TO 60
      DO 50 I=K,KK
         DO 50 J=L,LL
            DO 50 JA=1,KM1
               C(I,J) = C(I,J) - A(I,JA)*C(JA,J)
   50 CONTINUE
   60 IF (DL .EQ. 2) GO TO 80
      IF (DK .EQ. 2) GO TO 70
      T(1,1) = A(K,K) + B(L,L)
      IF (T(1,1) .EQ. 0.) RETURN
      C(K,L) = C(K,L) / T(1,1)
      IF (ABS(C(K,L)) .GE. RMAX) RETURN
      GO TO 100
   70 T(1,1) = A(K,K) + B(L,L)
      T(1,2) = A(K,KK)
      T(2,1) = A(KK,K)
      T(2,2) = A(KK,KK) + B(L,L)
      P(I) = C(K,L)
```

```
      P(2) = C(KK,L)
      NSYS = 2
      CALL SYSSLV
      IF (SING) RETURN
      C(K,L) = P(1)
      IF (ABS(C(K,L)) .GE. RMAX) RETURN
      C(KK,L) = P(2)
      IF (ABS(C(KK,L)) .GE. RMAX) RETURN
      GO TO 100
80    IF (DK .EQ.2) GO TO 90
      T(1,1) = A(K,K) + B(L,L)
      T(1,2) = B(LL,L)
      T(2,1) = B(L,LL)
      T(2,2) = A(K,K) + B(LL,LL)
      P(1) = C(K,L)
      P(2) = C(K,LL)
      NSYS = 2
      CALL SYSSLV
      IF (SING) RETURN
      C(K,L) = P(1)
      IF (ABS(C(K,L)) .GE. RMAX) RETURN
      C(K,LL) = P(2)
      IF (ABS(C(K,LL)) .GE. RMAX) RETURN
      GO TO 100
90    T(1,1) = A(K,K) + B(L,L)
      T(1,2) = A(K,KK)
      T(1,3) = B(LL,L)
      T(1,4) = 0.
      T(2,1) = A(KK,K)
      T(2,2) = A(KK,KK) + B(L,L)
      T(2,3) = 0.
      T(2,4) = T(1,3)
      T(3,1) = B(L,LL)
      T(3,2) = 0.
      T(3,3) = A(K,K) + B(LL,LL)
      T(3,4) = T(1,2)
      T(4,1) = 0.
      T(4,2) = T(3,1)
      T(4,3) = T(2,1)
      T(4,4) = A(KK,KK) + B(LL,LL)
      P(1) = C(K,L)
      P(2) = C(KK,L)
      P(3) = C(K,LL)
      P(4) = C(KK,LL)
      NSYS = 4
      CALL SYSSLV
      IF (SING) RETURN
      C(K,L) = P(1)
      IF (ABS(C(K,L)) .GE. RMAX) RETURN
      C(KK,L) = P(2)
      IF (ABS(C(KK,L)) .GE. RMAX) RETURN
      C(K,LL) = P(3)
      IF (ABS(C(K,LL)) .GE. RMAX) RETURN
      C(KK,LL) = P(4)
      IF (ABS(C(KK,LL)) .GE. RMAX) RETURN
100   K = K + DK
      IF (K .LE. M) GO TO 40
      L = L + DL
      IF (L .LE. N) GO TO 10
      FAIL = .FALSE.
      RETURN
      END
```

```
      SUBROUTINE SYSSLV
C
C     SYSSLV IS A FORTRAN IV SUBROUTINE THAT SOLVES THE LINEAR
C     SYSTEM AX = B OF ORDER N LESS THAN 5 BY CROUT REDUCTION
C     FOLLOWED BY BLOCK SUBSTITUTION.  THE MATRIX A, THE VECTOR B,
C     AND THE ORDER N ARE CONTAINED IN THE ARRAYS A,B, AND THE
C     VARIABLE N OF THE COMMON BLOCK SLVBLK.  THE SOLUTION IS RETURNED
C     IN THE ARRAY B.
C
      COMMON /SLVBLK/ A(5,5), B(5), N, SING
      REAL MAX
      LOGICAL SING
      SING = .TRUE.
    1 NM1 = N - 1
      N1 = N + 1
C
C
C     COMPUTE THE LU FACTORIZATION OF A
C
      DO 80 K=1,N
         KM1 = K-1
         IF (K .EQ. 1) GO TO 20
         DO 10 I=K,N
            DO 10 J=1,KM1
               A(I,K) = A(I,K) - A(I,J)*A(J,K)
   10    CONTINUE
   20    IF (K .EQ. N) GO TO 100
         KP1 = K+1
         MAX = ABS(A(K,K))
         INTR = K
         DO 30 I=KP1,N
            AA = ABS(A(I,K))
            IF (AA .LE. MAX) GO TO 30
            MAX = AA
            INTR = I
   30    CONTINUE
         IF (MAX .EQ. 0.) RETURN
         A(N1,K) = INTR
         IF (INTR .EQ. K) GO TO 50
         DO 40 J=1,N
            TEMP = A(K,J)
            A(K,J) = A(INTR,J)
            A(INTR,J) = TEMP
   40    CONTINUE
   50    DO 80 J=KP1,N
            IF (K .EQ. 1) GO TO 70
            DO 60 I=1,KM1
               A(K,J) = A(K,J) - A(K,I)*A(I,J)
   60       CONTINUE
   70       A(K,J) = A(K,J) / A(K,K)
   80 CONTINUE
C
C     INTERCHANGE THE COMPONENTS OF B
C
  100 DO 110 J=1,NM1
         INTR = A(N1,J)
         IF (INTR .EQ. J) GO TO 110
         TEMP = B(J)
         B(J) = B(INTR)
         B(INTR) = TEMP
  110 CONTINUE
C
C     SOLVE SY = B
C
  200 B(1) = B(1) / A(1,1)
      DO 220 I=2,N
         IM1 = I-1
         DO 210 J=1,IM1
            B(I) = B(I) - A(I,J)*B(J)
  210       CONTINUE
         B(I) = B(I) / A(I,I)
  220 CONTINUE
C
C     SOLVE UX=Y
```

```
C
300 DO 310 II=1,NM1
      I = NM1-II+1
      I1 = I+1
      DO 310 J=I1,N
         B(I) = B(I) - A(I,J)*B(J)
310 CONTINUE
      SING = .FALSE.
      RETURN
      END
```

```
      SUBROUTINE HQR3(A,V,N,NLOW,NUP,EPS,ER,EI,TYPE,NA,NV)
C
      INTEGER N,NA,NLOW,NUP,NV,TYPE(N)
      REAL A(NA,N),EI(N),ER(N),EPS,V(NV,N)
C
C     HQR3 REDUCES THE UPPER HESSENBERG MATRIX A TO QUASI-
C     TRIANGULAR FORM BY UNITARY SIMILARITY TRANSFORMATIONS.
C     THE EIGENVALUES OF A, WHICH ARE CONTAINED IN THE 1X1
C     AND 2X2 DIAGONAL BLOCKS OF THE REDUCED MATRIX, ARE
C     ORDERED IN DESCENDING ORDER OF MAGNITUDE ALONG THE
C     DIAGONAL.  THE TRANSFORMATIONS ARE ACCUMULATED IN THE
C     ARRAY V.  HQR3 REQUIRES THE SUBROUTINES EXCHNG,
C     QRSTEP, AND SPLIT.  THE PARAMETERS IN THE CALLING
C     SEQUENCE ARE (STARRED PARAMETERS ARE ALTERED BY THE
C     SUBROUTINE)
C
C        *A         AN ARRAY THAT INITIALLY CONTAINS THE N X N
C                   UPPER HESSENBERG MATRIX TO BE REDUCED.  ON
C                   RETURN A CONTAINS THE REDUCED, QUASI-
C                   TRIANGULAR MATRIX.
C        *V         AN ARRAY THAT CONTAINS A MATRIX INTO WHICH
C                   THE REDUCING TRANSFORMATIONS ARE TO BE
C                   MULTIPLIED.
C         N         THE ORDER OF THE MATRICES A AND V.
C         NLOW      A(NLOW-1,NLOW) AND A(NUP,NUP+1) ARE
C         NUP       ASSUMED TO BE ZERO, AND ONLY ROWS NLOW
C                   THROUGH NUP AND COLUMNS NLOW THROUGH
C                   NUP ARE TRANSFORMED, RESULTING IN THE
C                   CALCULATION OF EIGENVALUES NLOW
C                   THROUGH NUP.
C         EPS       A CONVERGENCE CRITERION.
C        *ER        AN ARRAY THAT ON RETURN CONTAINS THE REAL
C                   PARTS OF THE EIGENVALUES.
C        *EI        AN ARRAY THAT ON RETURN CONTAINS THE
C                   IMAGINARY PARTS OF THE EIGENVALUES.
C        *TYPE      AND INTEGER ARRAY WHOSE I-TH ENTRY IS
C                      0    IF THE I-TH EIGENVALUE IS REAL,
C                      1    IF THE I-TH EIGENVALUE IS COMPLEX
C                           WITH POSITIVE IMAGINARY PART.
C                      2    IF THE I-TH EIGENVALUE IS COMPLEX
C                           WITH NEGATIVE IMAGINARY PART,
C                     -1    IF THE I-TH EIGENVALUE WAS NOT
C                           CALCULATED SUCCESSFULLY.
C         NA        THE FIRST DIMENSION OF THE ARRAY A.
C         NV        THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES
C
      INTEGER I,IT,L,MU,NL,NU
      REAL E1,E2,P,Q,R,S,T,W,X,Y,Z
      LOGICAL FAIL
C
C     INITIALIZE.
C
      DO 10 I=NLOW,NUP
         TYPE(I) = -1
   10 CONTINUE
      T = 0.
C
C     MAIN LOOP. FIND AND ORDER EIGENVALUES.
C
      NU = NUP
  100 IF(NU .LT. NLOW) GO TO 500
         IT = 0
C
C        QR LOOP.  FIND NEGLIGABLE ELEMENTS AND PERFORM
C        QR STEPS.
C
  110    CONTINUE
C
C           SEARCH BACK FOR NEGLIGABLE ELEMENTS.
C
            L = NU
  120       CONTINUE
```

```
         IF(L .EQ. NLOW) GO TO 130
         IF(ABS(A(L,L-1)) .LT. EPS*(ABS(A(L-1,L-1))+ABS(A(L,L))))
     1      GO TO  130
         L = L-1
      GO TO 120
 130  CONTINUE
C
C     TEST TO SEE IF AN EIGENVALUE OR A 2X2 BLOCK
C     HAS BEEN FOUND.
C
      X = A(NU,NU)
      IF(L .EQ. NU) GO TO 300
      Y = A(NU-1,NU-1)
      W = A(NU,NU-1)*A(NU-1,NU)
      IF(L  .EQ. NU-1) GO TO 200
C
C     TEST ITERATION COUNT. IF IT IS 30 QUIT.  IF
C     IT IS 10 OR 20 SET UP AN AD-HOC SHIFT.
C
      IF(IT .EQ. 60) GO TO 500
      IF(IT.NE.40 .AND. IT.NE.50) GO TO 150
C
C        AD-HOC SHIFT.
C
         T = T + X
         DO 140 I=NLOW,NU
            A(I,I) = A(I,I) - X
 140     CONTINUE
         S = ABS(A(NU,NU-1)) + ABS(A(NU-1,NU-2))
         X = 0.75*S
         Y = X
         W = -0.4375*S**2
 150  CONTINUE
      IT = IT + 1
C
C     LOOK FOR TWO CONSECUTIVE SMALL SUB-DIAGONAL
C     ELEMENTS.
C
      NL= NU-2
 160  CONTINUE
         Z =  A(NL,NL)
         R = X - Z
         S = Y - Z
         P = (R*S-W)/A(NL+1,NL) + A(NL,NL+1)
         Q = A(NL+1,NL+1) - Z - R - S
         R = A(NL+2,NL+1)
         S = ABS(P) + ABS(Q) + ABS(R)
         P = P/S
         Q = Q/S
         R = R/S
         IF(NL .EQ. L) GO TO 170
         IF(ABS(A(NL,NL-1))*(ABS(Q)+ABS(R))  .LE.
     1      EPS*ABS(P)*(ABS(A(NL-1,NL-1))+ABS(Z)+ABS(A(NL+1,NL+1))))
     2      GO TO 170
         NL = NL-1
      GO TO 160
 170  CONTINUE
C
C     PERFORM A QR STEP BETWEEN NL AND NU.
C
      CALL QRSTEP(A,V,P,Q,R,NL,NU,N,NA,NV)
     GO TO 110
C
C    2X2 BLOCK FOUND.
C
 200 IF(NU .NE. NLOW+1) A(NU-1,NU-2) = 0.
     A(NU,NU) = A(NU,NU) + T
     A(NU-1,NU-1) = A(NU-1,NU-1) + T
     TYPE(NU) = 0
     TYPE(NU-1) = 0
     MU = NU
C
C    LOOP TO POSITION  2X2 BLOCK.
C
 210 CONTINUE
        NL = MU-1
```

```
               CALL EXCHNG(A,V,N,MU,1,2,EPS,FAIL,NA,NV)
            IF(.NOT. FAIL) GO TO 325
               TYPE(MU) = -1
               TYPE(MU+1) = -1
               TYPE(MU+2) = -1
               GO TO 500
  325       CONTINUE
               MU = MU+2
               GO TO 340
  330       CONTINUE
C
C              THE NEXT BLOCK IS 1X1.
C
               IF(ABS(A(MU,MU)) .GE. ABS(A(MU+1,MU+1)))
     1            GO TO 350
               CALL EXCHNG(A,V,N,MU,1,1,EPS,FAIL,NA,NV)
               MU = MU+1
  340       CONTINUE
         GO TO 320
  350    CONTINUE
         MU = NL
         NL = 0
      IF(MU .NE. 0) GO TO 310
C
C     GO BACK AND GET THE NEXT EIGFNVALUE.
C
  400 CONTINUE
      NU = L-1
      GO TO 100
C
C     ALL THE EIGNVALUES HAVE BEEN FOUND AND ORDERED.
C     COMPUTE THEIR VALUES AND TYPE.
C
  500 IF(NU .LT. NLOW) GO TO 507
      DO 503 I=1,NU
         A(I,I) = A(I,I) + T
  503 CONTINUE
  507 CONTINUE
      NU = NUP
  510 CONTINUE
      IF(TYPE(NU) .NE. -1) GO TO 515
         NU = NU-1
         GO TO 540
  515 CONTINUE
      IF(NU .EQ. NLOW) GO TO 520
      IF(A(NU,NU-1) .EQ. 0.) GO TO 520
C
C        2X2 BLOCK.
C
         CALL SPLIT(A,V,N,NU-1,E1,E2,NA,NV)
         IF(A(NU,NU-1) .EQ. 0.) GO TO 520
         ER(NU) = E1
         EI(NU-1) = E2
         ER(NU-1) = ER(NU)
         EI(NU) = -EI(NU-1)
         TYPE(NU-1) = 1
         TYPE(NU) = 2
         NU = NU-2
         GO TO 530
  520 CONTINUE
C
C        SINGLE ROOT.
C
         ER(NU) = A(NU,NU)
         EI(NU) = 0.
         NU = NU-1
  530 CONTINUE
  540 CONTINUE
      IF(NU .GE. NLOW) GO TO 510
      RETURN
      END
```

```
C
C       ATTEMPT  TO SPLIT THE BLOCK INTO TWO REAL
C       EIGENVALUES.
C
        CALL SPLIT(A,V,N,NL,E1,E2,NA,NV)
C
C       IF THE SPLIT WAS SUCCESSFUL, GO AND ORDER THE
C       REAL EIGENVALUES.
C
        IF(A(MU,MU-1) .EQ. 0.) GO TO 310
C
C       TEST TO SEE IF THE BLOCK IS PROPERLY POSITIONED,
C       AND IF NOT EXCHANGE IT
C
        IF(MU .EQ. NUP) GO TO 400
        IF(MU .EQ. NUP-1) GO TO 220
        IF(A(MU+2,MU+1) .EQ. 0.) GO TO 220
C
C         THE NEXT BLOCK IS 2X2.
C
          IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)
     1       .GE. A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*
     2            A(MU+2,MU+1))
     3      GO TO 400
          CALL EXCHNG(A,V,N,NL,2,2,EPS,FAIL,NA,NV)
          IF(.NOT. FAIL) GO TO 215
            TYPE(NL) = -1
            TYPE(NL+1) = -1
            TYPE(NL+2) = -1
            TYPE(NL+3) = -1
            GO TO 500
  215     CONTINUE
          MU = MU+2
          GO TO 230
  220   CONTINUE
C
C         THE NEXT BLOCK IS 1X1.
C
          IF(A(MU-1,MU-1)*A(MU,MU)-A(MU-1,MU)*A(MU,MU-1)
     1       .GE. A(MU+1,MU+1)**2)
     2      GO TO 400
          CALL EXCHNG(A,V,N,NL,2,1,EPS,FAIL,NA,NV)
          IF(.NOT. FAIL) GO TO 225
            TYPE(NL) = -1
            TYPE(NL+1) = -1
            TYPE(NL+2) = -1
            GO TO 500
  225     CONTINUE
          MU = MU+1
  230   CONTINUE
      GO TO 210
C
C     SINGLE EIGENVALUE FOUND.
C
  300 NL = 0
      A(NU,NU) = A(NU,NU) + T
      IF(NU .NE. NLOW) A(NU,NU-1) = 0.
      TYPE(NU) = 0
      MU = NU
C
C     LOOP TO POSITION ONE OR TWO REAL EIGENVALUES.
C
  310 CONTINUE
C
C       POSITION THE EIGENVALUE LOCATED AT A(NL,NL).
C
  320   CONTINUE
          IF(MU .EQ. NUP) GO TO 350
          IF(MU .EQ. NUP-1) GO TO 330
          IF(A(MU+2,MU+1) .EQ. 0.) GO TO 330
C
C           THE NEXT BLOCK IS 2X2.
C
            IF(A(MU,MU)**2 .GE.
     1         A(MU+1,MU+1)*A(MU+2,MU+2)-A(MU+1,MU+2)*A(MU+2,MU+1))
     2        GO TO 400
```

```
      SUBROUTINE EXCHNG(A,V,N,L,B1,B2,EPS,FAIL,NA,NV)
C
      INTEGER B1,B2,L,NA,NV
      REAL A(NA,N),EPS,V(NV,N)
      LOGICAL FAIL
C
C     GIVEN THE UPPER HESSENBERG MATRIX A WITH CONSECUTIVE
C     B1XB1 AND B2XB2 DIAGONAL BLOCKS (B1,B2 .LE. 2)
C     STARTING AT A(L,L), EXCHNG PRODUCES A UNITARY
C     SIMILARITY TRANSFORMATION THAT EXCHANGES THE BLOCKS
C     ALONG WITH THEIR EIGENVALUES.  THE TRANSFORMATION
C     IS ACCUMULATED IN V.  EXCHNG REQUIRES THE SUBROUTINE
C     QRSTEP.  THE PARAMETERS IN THE CALLING SEQUENCE ARE
C     (STARRED PARAMETERS ARE ALTERED BY THE SUBROUTINE)
C
C        *A        THE MATRIX WHOSE BLOCKS ARE TO BE
C                  INTERCHANGED.
C        *V        THE ARRAY INTO WHICH THE TRANSFORMATIONS
C                  ARE TO BE ACCUMULATED.
C         N        THE ORDER OF THE MATRIX A.
C         L        THE POSITION OF THE BLOCKS.
C         B1       THE SIZE OF THE FIRST BLOCK.
C         B2       THE SIZE OF THE SECOND BLOCK.
C         EPS      A CONVERGENCE CRITERION.
C        *FAIL     A LOGICAL VARIABLE WHICH IS FALSE ON A
C                  NORMAL RETURN.  IF THIRTY ITERATIONS WERE
C                  PERFORMED WITHOUT CONVERGENCE, FAIL IS SET
C                  TO TRUE AND THE ELEMENT
C                  A(L+B2,L+B2-1) CANNOT BE ASSUMED ZERO.
C         NA       THE FIRST DIMENSION OF THE ARRAY A.
C         NV       THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES.
C
      INTEGER I,IT,J,L1,M
      REAL P,Q,R,S,W,X,Y,Z
C
      FAIL = .FALSE.
      IF(B1 .EQ. 2) GO TO 40
        IF(B2 .EQ. 2) GO TO 10
C
C           INTERCHANGE 1X1 AND 1X1 BLOCKS.
C
          L1 = L+1
          Q = A(L+1,L+1) - A(L,L)
          P = A(L,L+1)
          R = AMAX1(P,Q)
          IF(R .EQ. 0.) RETURN
          P = P/R
          Q = Q/R
          R = SQRT(P**2 + Q**2)
          P = P/R
          Q = Q/R
          DO 3 J=L,N
            S = P*A(L,J) + Q*A(L+1,J)
            A(L+1,J) = P*A(L+1,J) - Q*A(L,J)
            A(L,J) = S
    3     CONTINUE
          DO 5 I=1,L1
            S = P*A(I,L) + Q*A(I,L+1)
            A(I,L+1) = P*A(I,L+1) - Q*A(I,L)
            A(I,L) = S
    5     CONTINUE
          DO 7 I=1,N
            S = P*V(I,L) + Q*V(I,L+1)
            V(I,L+1) = P*V(I,L+1) - Q*V(I,L)
            V(I,L) = S
    7     CONTINUE
          A(L+1,L) = 0.
          RETURN
   10   CONTINUE
C
C           INTERCHANGE 1X1 AND 2X2 BLOCKS.
C
```

```
      X = A(L,L)
      P = 1.
      Q = 1.
      R = 1.
      CALL QRSTEP(A,V,P,Q,R,L,L+2,N,NA,NV)
      IT = 0
20    IT = IT+1
        IF(IT .LE. 60) GO TO 30
          FAIL = .TRUE.
          RETURN
30      CONTINUE
        P = A(L,L) - X
        Q = A(L+1,L)
        R = 0.
        CALL QRSTEP(A,V,P,Q,R,L,L+2,N,NA,NV)
        IF(ABS(A(L+2,L+1)) .GT.
     1                         EPS*(ABS(A(L+1,L+1))+ABS(A(L+2,L+2))))
     1    GO TO 20
      A(L+2,L+1) = 0.
      RETURN
40 CONTINUE
C
C     INTERCHANGE 2X2 AND B2XB2 BLOCKS.
C
      M = L+2
      IF(B2 .EQ. 2) M = M+1
      X = A(L+1,L+1)
      Y = A(L,L)
      W = A(L+1,L)*A(L,L+1)
      P = 1.
      Q = 1.
      R = 1.
      CALL QRSTEP(A,V,P,Q,R,L,M,N,NA,NV)
      IT = 0
50    IT = IT+1
        IF(IT .LE. 60) GO TO 60
          FAIL = .TRUE.
          RETURN
60      CONTINUE
        Z = A(L,L)
        R = X - Z
        S = Y - Z
        P = (R*S-W)/A(L+1,L) + A(L,L+1)
        Q = A(L+1,L+1) - Z - R - S
        R = A(L+2,L+1)
        S = ABS(P) + ABS(Q) + ABS(R)
        P = P/S
        Q = Q/S
        R = R/S
        CALL QRSTEP(A,V,P,Q,R,L,M,N,NA,NV)
      IF(ABS(A(M-1,M-2)) .GT. EPS*(ABS(A(M-1,M-1))+ABS(A(M-2,M-2))))
     1  GO TO 50
      A(M-1,M-2) = 0.
      RETURN
    CONTINUE
    END
```

```
      SUBROUTINE SPLIT(A,V,N,L,E1,E2,NA,NV)
C
      INTEGER L,N,NA,NV
      REAL A(NA,N),V(NV,N)
C
C     GIVEN THE UPPER HESSENBERG MATRIX A WITH A 2X2 BLOCK
C     STARTING AT A(L,L), SPLIT DETERMINES IF THE
C     CORRESPONDING EIGENVALUES ARE REAL OR COMPLEX. IF THEY
C     ARE REAL, A ROTATION IS DETERMINED THAT REDUCES THE
C     BLOCK TO UPPER TRIANGULAR FORM WITH THE EIGENVALUE
C     OF LARGEST ABSOLUTE VALUE APPEARING FIRST.  THE
C     ROTATION IS ACCUMULATED IN V.  THE EIGENVALUES (REAL
C     OR COMPLEX) ARE RETURNED IN E1 AND E2.  THE PARAMETERS
C     IN THE CALLING SEQUENCE ARE (STARRED PARAMETERS ARE
C     ALTERED BY THE SUBROUTINE)
C
C        *A          THE UPPER HESSENVERG MATRIX WHOSE 2X2
C                    BLOCK IS TO BE SPLIT.
C        *V          THE ARRAY IN WHICH THE SPLITTING TRANS-
C                    FORMATION IS TO BE ACCUMULATED.
C         N          THE ORDER OF THE MATRIX A.
C         L          THE POSITION OF THE 2X2 BLOCK.
C        *E1         ON RETURN IF THE EIGENVALUES ARE COMPLEX
C        *E2         E1 CONTAINS THEIR COMMON REAL PART AND
C                    E2 CONTAINS THE POSITIVE IMAGINARY PART.
C                    IF THE EIGENVALUES ARE REAL, E1 CONTAINS
C                    THE ONE LARGEST IN ABSOLUTE VALUE AND E2
C                    CONTAINS THE OTHER ONE.
C        NA          THE FIRST DIMENSION OF THE ARRAY A.
C        NV          THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES
C
      INTEGER I,J,L1
      REAL P,Q,R,T,U,W,X,Y,Z
C
      X = A(L+1,L+1)
      Y = A(L,L)
      W = A(L,L+1)*A(L+1,L)
      P = (Y-X)/2.
      Q = P**2 + W
      IF(Q .GE. 0.) GO TO 5
C
C        COMPLEX EIGENVALUE.
C
         E1 = P + X
         E2 = SQRT(-Q)
         RETURN
    5 CONTINUE
C
C     TWO REAL EIGENVALUES.  SET UP TRANSFORMATION.
C
      Z = SQRT(Q)
      IF(P .LT. 0.) GO TO 10
         Z = P + Z
         GO TO 20
   10 CONTINUE
         Z = P - Z
   20 CONTINUE
      IF(Z .EQ. 0.) GO TO 30
         R = -W/Z
         GO TO 40
   30 CONTINUE
         R = 0.
   40 CONTINUE
      IF(ABS(X+Z) .GE. ABS(X+R)) Z = R
      Y = Y - X - Z
      X = -Z
      T = A(L,L+1)
      U = A(L+1,L)
      IF(ABS(Y)+ABS(U) .LE. ABS(T)+ABS(X)) GO TO 60
         Q = U
         P = Y
         GO TO 70
```

```
60 CONTINUE
      Q = X
      P = T
70 CONTINUE
   R = SQRT(P**2 + Q**2)
   IF(R .GT. 0.) GO TO 80
   E1 = A(L,L)
   E2 = A(L+1,L+1)
      A(L+1,L) = 0.
      RETURN
80 CONTINUE
   P = P/R
   Q = Q/R
C
C     PREMULTIPLY.
C
   DO 90 J=L,N
      Z = A(L,J)
      A(L,J) = P*Z + Q*A(L+1,J)
      A(L+1,J) = P*A(L+1,J) - Q*Z
90 CONTINUE
C
C     POSTMULTIPLY.
C
   L1 = L+1
   DO 100 I=1,L1
      Z = A(I,L)
      A(I,L) = P*Z + Q*A(I,L+1)
      A(I,L+1) = P*A(I,L+1) - Q*Z
100 CONTINUE
C
C     ACCUMULATE THE TRANSFORMATION IN V.
C
   DO 110 I=1,N
      Z = V(I,L)
      V(I,L) = P*Z + Q*V(I,L+1)
      V(I,L+1) = P*V(I,L+1) - Q*Z
110 CONTINUE
   A(L+1,L) = 0.
   E1 = A(L,L)
   E2 = A(L+1,L+1)
   RETURN
   END
```

```
      SUBROUTINE QRSTEP(A,V,P,Q,R,NL,NU,N,NA,NV)
C
      INTEGER N,NA,NL,NU,NV
      REAL A(NA,N),P,Q,R,V(NV,N)
C
C     QRSTEP PERFORMS ONE IMPLICIT QR STEP ON THE
C     UPPER HESSENBERG MATRIX A.  THE SHIFT IS DETERMINED
C     BY THE NUMBERS P,Q, AND R, AND THE STEP IS APPLIED TO
C     ROWS AND COLUMNS NL THROUGH NU.  THE TRANSFORMATIONS
C     ARE ACCUMULATED IN V.  THE PARAMETERS IN THE CALLING
C     SEQUENCE ARE (STARRED APRAMETERS ARE ALTERED BY THE
C     SUBROUTINE)
C
C        *A          THE UPPER HESSENBERG MATRIX ON WHICH THE
C                    QR STEP IS TO BE PERFORMED.
C        *V          THE ARRAY IN WHICH THE TRANSFORMATIONS
C                    ARE TO BE ACCUMULATED
C        *P          PARAMETERS THAT DETERMINE THE SHIFT.
C        *Q
C        *R
C         NL         THE LOWER LIMIT OF THE STEP.
C         NU         THE UPPER LIMIT OF THE STEP.
C         N          THE ORDER OF THE MATRIX A.
C         NA         THE FIRST DIMENSION OF THE ARRAY A.
C         NV         THE FIRST DIMENSION OF THE ARRAY V.
C
C     INTERNAL VARIABLES.
C
      INTEGER I,J,K,NL2,NL3,NUM1
      REAL S,X,Y,Z
      LOGICAL LAST
C
      NL2 = NL+2
      DO 10 I=NL2,NU
         A(I,I-2) = 0.
   10 CONTINUE
      IF(NL2 .EQ. NU) GO TO 30
         NL3 = NL+3
         DO 20 I=NL3,NU
            A(I,I-3) = 0.
   20    CONTINUE
   30 CONTINUE
      NUM1 = NU-1
      DO 130 K=NL,NUM1
C
C        DETERMINE THE TRANSFORMATION.
C
         LAST = K .EQ. NUM1
         IF(K .EQ. NL) GO TO 40
            P = A(K,K-1)
            Q = A(K+1,K-1)
            R = 0.
            IF(.NOT.LAST) R = A(K+2,K-1)
            X = ABS(P) + ABS(Q) + ABS(R)
            IF(X .EQ. 0.) GO TO 130
            P = P/X
            Q = Q/X
            R = R/X
   40    CONTINUE
         S = SQRT(P**2 + Q**2 + R**2)
         IF(P .LT. 0.) S = -S
         IF(K .EQ. NL) GO TO 50
            A(K,K-1) = -S*X
            GO TO 60
   50    CONTINUE
            IF(NL .NE. 1) A(K,K-1) = -A(K,K-1)
   60    CONTINUE
         P = P + S
         X = P/S
         Y = Q/S
         Z = R/S
         Q = Q/P
         R = R/P
C
```

```
C       PREMULTIPLY.
C
        DO 80 J=K,N
          P = A(K,J) + Q*A(K+1,J)
          IF(LAST) GO TO 70
            P = P + R*A(K+2,J)
            A(K+2,J) = A(K+2,J) - P*Z
   70     CONTINUE
          A(K+1,J) = A(K+1,J) - P*Y
          A(K,J) = A(K,J) - P*X
   80   CONTINUE
C
C       POSTMULTIPLY.
C
        J = MIN0(K+3,NU)
        DO 100 I=1,J
          P = X*A(I,K) + Y*A(I,K+1)
          IF(LAST) GO TO 90
            P = P + Z*A(I,K+2)
            A(I,K+2) = A(I,K+2) - P*R
   90     CONTINUE
          A(I,K+1) = A(I,K+1) - P*Q
          A(I,K) = A(I,K) - P
  100   CONTINUE
C
C       ACCUMULATE THE TRANSFORMATION IN V.
C
        DO 120 I=1,N
          P = X*V(I,K) + Y*V(I,K+1)
          IF(LAST) GO TO 110
            P = P + Z*V(I,K+2)
            V(I,K+2) = V(I,K+2) - P*R
  110     CONTINUE
          V(I,K+1) = V(I,K+1) - P*Q
          V(I,K) = V(I,K) - P
  120   CONTINUE
  130 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE ORTHES(NM,N,LOW,IGH,A,ORT)
      INTEGER I,J,M,N,II,JJ,LA,MP,NM,IGH,KP1,LOW
      REAL A(NM,N),ORT(IGH)
      REAL F,G,H,SCALE
      LA = IGH - 1
      KP1 = LOW + 1
      IF(LA .LT. KP1) GO TO 200
      DO 180 M=KP1,LA
         H = 0.
         ORT(M) = 0.
         SCALE = 0.
         DO 90 I=1,IGH
   90    SCALE = SCALE + ABS(A(I,M-1))
         IF(SCALE .EQ. 0.) GO TO 180
         MP = M + IGH
         DO 100 II=M,IGH
            I = MP - II
            ORT(I) = A(I,M-1)/SCALE
            H = H + ORT(I)*ORT(I)
  100    CONTINUE
         G = -SIGN(SQRT(H),ORT(M))
         H = H- ORT(M)*G
         ORT(M) = ORT(M) - G
         DO 130 J=M,N
            F = 0.
            DO 110 II=M,IGH
               I = MP - II
               F = F + ORT(I)*A(I,J)
  110       CONTINUE
      IF (H .EQ.0.) GO TO 162
            F = F/H
            DO 120 I=M,IGH
               A(I,J) = A(I,J) - F*ORT(I)
  120       CONTINUE
  130    CONTINUE
         DO 160 I=1,IGH
            F = 0.
            DO 140 JJ=M,IGH
               J = MP - JJ
               F = F + ORT(J)*A(I,J)
  140       CONTINUE
            F = F/H
            DO 150 J=M,IGH
               A(I,J) = A(I,J) - F*ORT(J)
  150       CONTINUE
  160    CONTINUE
  162 CONTINUE
         ORT(M) = SCALE*ORT(M)
         A(M,M-1) = SCALE*G
  180 CONTINUE
  200 RETURN
      END
```

```
      SUBROUTINE ORTRAN(NM,N,LOW,IGH,A,ORT,Z)
      INTEGER I,J,N,KL,MM,MP,NM,IGH,LOW,MP1
      REAL A(NM,IGH),ORT(IGH),Z(NM,N)
      REAL G,H
      DO 80 I=1,N
         DO 60 J=1,N
            Z(I,J) = 0.
   60    CONTINUE
         Z(I,I) = 1.
   80 CONTINUE
      KL = IGH - LOW - 1
      IF(KL .LT. 1) GO TO 200
      DO 140 MM=1,KL
         MP = IGH - MM
         H = A(MP,MP-1)*ORT(MP)
         IF(H .EQ. 0.) GO TO 140
         MP1 = MP+1
         DO 100 I=MP1,IGH
            ORT(I) = A(I,MP-1)
  100    CONTINUE
         DO 130 J=MP,IGH
            G = 0.
            DO 110 I=MP,IGH
               G = G + ORT(I)*Z(I,J)
  110       CONTINUE
            G = G/H
            DO 120 I=MP,IGH
               Z(I,J) = Z(I,J) + G*ORT(I)
  120       CONTINUE
  130    CONTINUE
  140 CONTINUE
  200 RETURN
      END
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER ONR-N00014-76-C-0391-489	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 AN ALGORITHM FOR COMPUTING REDUCING SUBSPACES BY BLOCK DIAGONALIZATION.		5. TYPE OF REPORT & PERIOD COVERED Technical Report.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Connice A. Bavely [redacted] G. W. Stewart		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-76-C-0391
9. PERFORMING ORGANIZATION NAME AND ADDRESS Department of Computer Science University of Maryland College Park, Maryland 20742		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 46p.
11. CONTROLLING OFFICE NAME AND ADDRESS Mathematics Branch Office of Naval Research Arlington, VA 22217		12. REPORT DATE 11 Oct[redacted]76
		13. NUMBER OF PAGES 45
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) 14 TR-489		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

reducing subspaces
block diagonalization
eigenvalues
eigenvectors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper describes an algorithm for reducing a real matrix A to block diagonal form by a real similarity transformation. The columns of the transformation corresponding to a block span a reducing subspace of A, and the block is the representation of A in that subspace with respect to the basis. The algorithm attempts to control the condition of the transformation matrices, so that the reducing subspaces are well conditioned and the basis vectors are numerically independent.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

409022

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)